# Graph based physical models for sound synthesis

Christensen, Pelle Juul; Serafin, Stefania

*Published in:*
Proceedings of the 16th Sound and Music Computing Conference, SMC 2019



*Publication date:*
2019

*Document Version*
Publisher's PDF, also known as Version of record

# Graph Based Physical Models for Sound Synthesis

**Pelle Juul Christensen**
Aalborg University
Copenhagen, Denmark
pelle.juul@tuta.io

**Stefania Serafin**
Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark
sts@create.aau.dk

## ABSTRACT

We focus on physical models in which multiple strings are connected via junctions to form graphs. Starting with the case of the 1D wave equation, we show how to extend it to a string branching into two other strings, and from there how to build complex cyclic and acyclic graphs. We introduce the concept of dense models and show that a discretization of the 2D wave equation can be built using our methods, and that there are more efficient ways of modelling 2D wave propagation than a rectangular grid. We discuss how to apply Dirichlet and Neumann boundary conditions to a graph model, and show how to compute the frequency content of a graph using common methods. We then prove general lower and upper bounds computational complexity. Lastly, we show how to extend our results to other kinds of acoustical objects, such as linear bars, and how to add dampening to a graph model. A reference implementation in MATLAB and an interactive JUCE/C++ application is available online.

## 1. INTRODUCTION

Recent research in physical models has been directed towards simulating systems using *finite difference schemes* [1], which can be used to model the intricacies of many kinds of vibrating systems and exciters.

As many other physical modelling methods, finite difference schemes can be used to simulate systems which are hard to construct in real life. For example, we can tweak the parameters of models to make e.g., strings that are extremely long or violin bows that move faster than the human anatomy allows for. Some research has explored this idea further by building abstract physical models that do not have any direct relation to any real world instruments. For example, the work of Stefan Bilbao includes ways of constructing modular percussion instruments by connecting vibrating bars and plates [2]. Similarly, the CORDIS-ANIMA project of ACROE allows one to build virtual instruments by combining masses, springs, friction elements and non-linear links, to create novel compositions and matching animations [3].



This paper will further explore the notions of abstract physical models by showing a way to build systems of connected strings that would not necessarily be achievable or practical in real life. The models described in this paper could possiblly be constructed physically; the goal of the current study is, however, to build synthesis algorithms inspired by physical phenomena but without attention to whether they are realisable in real life or not.

While we are looking at finite difference schemes we should be aware that many methods of physical modelling for sound synthesis exists and that they are largely equivalent with regards to which sounds we can produce using them. Related to the work at hand are waveguides, modal synthesis and mass-spring systems [4] [5], [6]. For a nice description and discussion of the various methods see [1, Chapter 1].

We start in Section and 2 by reviewing the one-dimensional wave equation and how to make a finite-difference scheme to simulate it. In Section 3 we take a derivation of the regular 1D wave equation, and show how to get similar results for a branching topology. In Section 4 we see how to use these results to build various kinds of models. In Section 5 we investigate the properties of *dense models*, and show that a special case of these is equivalent to a discretization the 2D wave equation. In section 6 we deal with the boundary conditions of our models. Next, in Section 7 we discuss the computational complexity of various models, and in Section 8 we give a method of computing natural mode frequencies and shapes. Lastly in Sections 10, 11 and 12 we end with a description of the reference implementation, suggestions for future work, and concluding remarks.

## 2. THE 1D WAVE EQUATION

Before building complex abstract models, we will review the 1D wave equation, which will be used as the starting point for further exploration, since it is thoroughly studied and perhaps the simplest spatial model of musical utility. This section, provided for completeness, is completely textbook and based on [2], [1], [7], [8], and [9].

The 1D wave equation is defined by the second-order partial differential equation

$$u_{tt} = c^2 u_{xx}, \tag{1}$$

where $u = u(x, t)$ is a variable describing the deformation of the medium at position $x \in [0, 1]$ and time $t \in [0, \infty]$. The constant $c$ is the normalized wave speed which is determined by the medium under consideration. When dis-

cretized, Equation (1) looks like

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n, \tag{2}$$

where the finite difference operators $\delta_{tt}$ and $\delta_{xx}$ are defined as

$$\delta_{tt} u_l^n = \frac{1}{k^2}(u_l^{n+1} - 2u_l^n + u_l^{n-1}) \approx \frac{d^2u}{dt^2}, \tag{3}$$

$$\delta_{xx} u_l^n = \frac{1}{h^2}(u_{l+1}^n - 2u_l^n + u_{l+1}^n) \approx \frac{d^2u}{dt^2}. \tag{4}$$

To implement this model, we expand the temporal operator in Equation (2) and isolate $u_l^{n+1}$ to get

$$u_l^{n+1} = k^2 c^2 \delta_{xx} u_l^n + 2u_l^n - u_l^{n-1}, \tag{5}$$

## 3. TWO-BRANCH AND N-BRANCH TOPOLOGIES

The 1D wave equation can be considered an idealization of a real physical string (e.g. a guitar string) under low amplitude conditions. We can represent this using a graph with two nodes and one edge, as seen in Figure. 1. a).

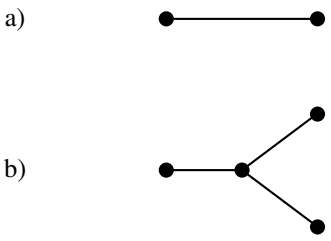


Figure 1. a) A line topology model — a string viewed as a graph with two nodes and one edge. b) A branching topology model — one string branching out into two other strings.

Once we look at our system as a graph, a new perspective arises: if we can build this kind of graph, what other graphs can we create? In this section we will look at the case of a *branching* topology — one string segment connected to two other string segments through one node, as visualized in Figure 1. b). We will investigate how to model wave propagation on such a graph.

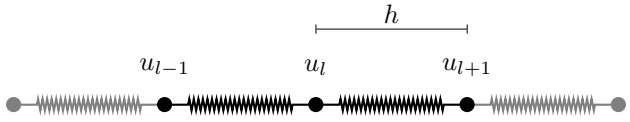


Figure 2. A section of a line topology string view as a lumped mass-spring network.

To understand the branching topology we must first look at the line topology more in-depth. We will be using a derivation found in [1, Chapter 6] and similarly in [8]. In Figure 2. we see a section of a string viewed as a network of masses connected via springs. The dynamics of the mass $u_l$ will be defined by the ordinary differential equation

$$m\frac{d^2u_l}{dt^2} = f_{l+1,l} - f_{l,l-1}, \tag{6}$$

where $m$ is the mass of the node and e.g. $f_{l+1,l}$ is defined by

$$f_{l+1,l} = \kappa(u_{l+1} - u_l), \tag{7}$$

which is the force caused by the spring between $u_{l+1}$ and $u_l$, where $\kappa$ is the spring constant.

Combining Equations (6) and (7) we get

$$m\frac{d^2u_l}{dt^2} = \kappa(u_{l+1} - 2u_l + u_{l-1}). \tag{8}$$

Defining $m = \rho A h$ where $\rho$ is density, $A$ the cross-sectional area of the string, and $h$ is the distance between the node. Then setting $\kappa = EA/h$, where $E$ is the Young's modulus of the material, we get

$$\frac{d^2u_l}{dt^2} = \frac{E}{\rho}\left(\frac{u_{l+1} - 2u_l + u_{l-1}}{h^2}\right). \tag{9}$$

Notice that the right-hand side of this equation is equivalent to $c\delta_{xx}u$ when $c = \sqrt{E/\rho}$.

Now we perform the same derivation, but for the branching topology. A mass-spring network of the branching point is shown in Figure 3. For simplicity and for the remainder of this paper we will assume that all connected strings has the same parameters ($h$, $k$ and $c$).

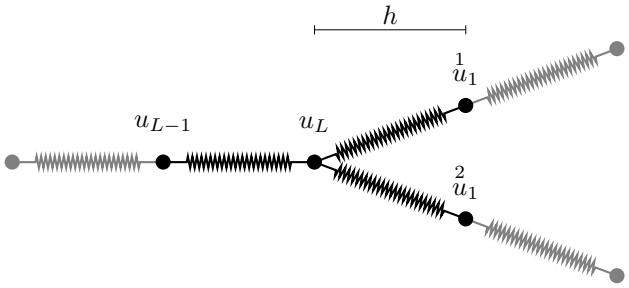


Figure 3. The branching point of a branching topology viewed as a mass-spring network.

The dynamics of $u_L$ is described by

$$m\frac{d^2u}{dt^2} = \overset{1}{f}_{1,L} + \overset{2}{f}_{1,L} - f_{L,L-1}. \tag{10}$$

Notice that we are using $L$ instead of $l$ since we are at the end of string segment $u$ and that we are using 1 instead of $l$ for $\overset{1}{u}$ and $\overset{2}{u}$ since we are at the beginning of those string segments.

The spring forces of the branching node is

$$\overset{1}{f}_{1,L} = \kappa(\overset{1}{u}_1 - u_L), \tag{11}$$

$$\overset{2}{f}_{1,L} = \kappa(\overset{2}{u}_1 - u_L). \tag{12}$$

Taking the same steps as we did to reach Equation (9) but using Equations (10) through (12) we get

$$\frac{d^2u_L}{dt^2} = \frac{E}{\rho}\left(\frac{\overset{1}{u}_1 + \overset{2}{u}_1 + u_{L-1} - 3u_L}{h^2}\right). \tag{13}$$

By analogy to Equation (9) we then have a definition for $\delta_{xx}u_L$ in the case of the branching topology, which is

$$\delta_{xx}u_L = \frac{1}{h^2}(\overset{1}{u}_1 + \overset{2}{u}_1 + u_{L-1} - 3u_L). \qquad (14)$$

Of course we are not limited to just two branches. We can create a N-branch topology as in Figure 4.

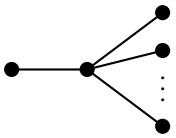

Figure 4. A model with an $N$-branch topology.

Using a derivation similar to that of the two-branch model, we get a definition of $\delta_{xx}u_L$ which looks like

$$\delta_{xx}u_L = \frac{1}{h^2}\left(\sum_{\overset{*}{u}\in\mathbb{U}_L} \overset{*}{u} - |\mathbb{U}_L|u_L\right), \qquad (15)$$

where $\mathbb{U}_L$ is the set of all the end nodes of the string segments connected to the branching node $u$ including $u_{L-1}$.

For now we have only considered a branch at the end of a string. We could just as well have considered a branch at the beginning and arrived at a result similar to Equation (15). We can generalize our rule such that we can apply that rule to any point in our graph, even the internal nodes:

$$\delta_{\Delta\mathbb{U}}u_l = \frac{1}{h^2}\left(\sum_{\overset{*}{u}\in\mathbb{U}_l} \overset{*}{u} - |\mathbb{U}_l|u_l\right). \qquad (16)$$

Notice that we have changed our notation to $\delta_{\Delta\mathbb{U}}$ instead of $\delta_{xx}$. This is to avoid confusion with the one-dimensional $\delta_{xx}$ and the $|\mathbb{U}_l|$ dimensional $\delta_{\Delta\mathbb{U}}$.

Note that the angle between the strings in a junction are not considered since the nodes have no freedom of movement in the 2D plane in which we are building our graphs. Also, when drawing a graph we will usually not care about getting the distances and angles right, what's important is how the strings connect and how many internal nodes they have.

The notion of a acoustic junctions is not a new concept in physical modelling. 2D and 3D finite difference schemes already contains scattering junctions arranged as grid [1, Chapter 6] [10], however the author has not seen the concept presented as in the current paper where the junctions do not need to be distributed homogeneously throughout the model. Likewise, the physical modeling method of *digital waveguides*, which has been proved equivalent to finite difference schemes [11], use scattering junctions to great extend in order to model room and instrument acoustics [12] [13].

## 4. BUILDING MODELS

Using Equation (16) we are able to connect any number of strings together. For example we could take a topology like Figure 1. b) and connect another layer to the rightmost nodes to get a topology like Figure 5.

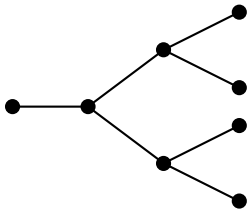

Figure 5. A model with a binary tree-like topology.

In graph theory, we would call such a graph a *tree*. More generally we can call it an *acyclic graph*. Any graph we can build by only adding strings to the end of other strings will be a tree. So far all models we have looked have been trees.

We can build more exciting graphs by connecting the ends of two strings using another string. For example, we can take the graph in Figure 5 and connect the top right node to the leftmost node, resulting in the graph in Figure 6, which is a *cyclic graph*.

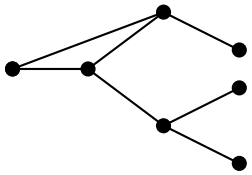

Figure 6. A graph with a single cycle

We may also take a string and connect its two ends, thus creating a loop as seen in Figure 7. This is a compelling case because it is a model that does not need any boundary conditions. In general, any graph without pendant nodes can be evaluated without boundary conditions. Physical models with looping topology has previously been studied in the case of Tibetan singing bowls and glass harmonicas [14].

Figure 7. A string with its two ends joined, forming a graph with a single cycle

## 5. DENSE TOPOLOGIES

A point that has been implicit so far is that each string in a given model must be assigned a number of internal nodes, just like we assign the ordinary 1D wave equation number of nodes when we discretize it.

If we construct a model in which there are no strings longer than one between each branching node, we say that the model is dense [1] .

[1] Note that some dense models may have string segments longer than one at the edges, we will ignore this fact for now since the internal structure of such a model will be dense

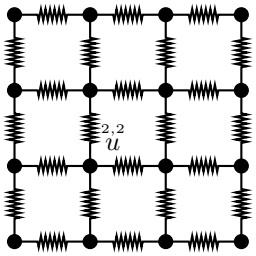


Figure 8. A dense rectangular grid. The zigzag lines, symbolising springs, are drawn to show that the model has no internal strings.

One well known dense model is the rectangular grid, as seen in Figure 8. When we wish to update one of the inner nodes $^{x,y}u$ we use Equation (16), which in this case takes the form

$$\delta_{xx}\,{}^{x,y}u = \frac{1}{h^2}\left({}^{x,y-1}u + {}^{x-1,y}u + {}^{x+1,y}u + {}^{x,y+1}u - 4\,{}^{x,y}u\right). \tag{17}$$

The two-dimensional version of the wave equation, which models wave propagation on a non-stiff membrane is defined by [1, Chapter 5]

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right). \tag{18}$$

A finite difference scheme for this could look like

$$\delta_{tt}\,{}^{x,y}u = c^2\left(\delta_{xx}\,{}^{x,y}u + \delta_{yy}\,{}^{x,y}u\right). \tag{19}$$

If we expand the spatial difference operators we get

$$\delta_{xx}\,{}^{x,y}u + \delta_{yy}\,{}^{x,y}u = \frac{1}{h^2}\left({}^{x,y-1}u + {}^{x-1,y}u + {}^{x+1,y}u + {}^{x,y+1}u - 4\,{}^{x,y}u\right), \tag{20}$$

which is equivalent to Equation (17). Therefore, the model in Figure 8. is equivalent to the discretized 2D wave equation.

However, since we are building grids using nodes and not from the definition of the 2D partial derivative, we can build grids which are not rectangular. For example, Figure 9. shows a hexagonal grid.

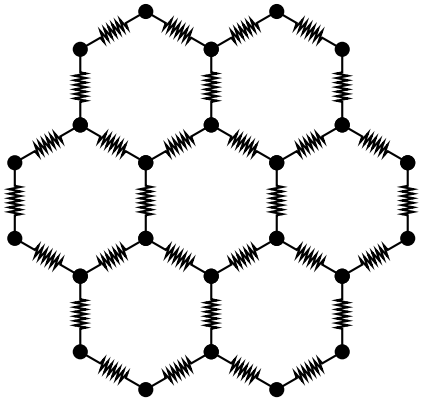

Figure 9. A dense hexagon grid.

Grid configurations are common throughout physical modeling literature. Especially the method of using digital waveguides has investigated various grid configurations and provide tools for building arbitrary typologies. Since Digital waveguides has as been proved equal to finite-difference schemes the result of those studies should be applicable to graph physical models as well [15] [5] [11].

Say we wanted to model the wave propagation on a 2D membrane, which kind of grid would be more efficient with respect to area covered? If we first look at the rectangular grid we see that we cover

$$R_4 = \frac{A}{N} = \frac{1}{4} \tag{21}$$

unit area per node. Where $A$ is the area of a grid unit, and $N$ is the number of nodes in each unit, considering only inner grid units.

If we do the same for the hexagonal grid we get

$$R_6 = \frac{A}{N} = \frac{3\sqrt{3}(1^2)/2}{6} = \frac{\sqrt{3}}{4}. \tag{22}$$

Since we have that

$$\frac{\sqrt{3}}{4} > \frac{1}{4}, \tag{23}$$

we can cover more area using the same amount of nodes when arranging them in a hexagonal grid, compared to a rectangular grid, giving us a more efficient model.

Furthermore, each node of an inner grid unit in the rectangular grid has connectivity 4 (each node connects to four other nodes), while the hexagon grid has connectivity 3. One can infer from the results of Section 7. that if we have the same amount of nodes, this causes the hexagon grid to have a lower computational complexity than the rectangular grid.

The differences between rectangular and hexagonal grids have been investigated before by Bilbao and Hamilton in [10], where they arrive at a conclusion similar the one presented here with the addition of also analysing the dispersion of each configuration.

Much is still left to be said about dense configurations. One does not have limit oneself to regular grids or grids at all. However, the nuances of various dense configurations are beyond the scope of the current project and could be enough work for a separate paper.

## 6. BOUNDARY CONDITIONS

To evaluate most models we need to decide on which boundary conditions to use for at least the pendant nodes.

When considering a string topology model we need only apply boundary conditions at the two ends. However, since graph models can have multiple pendant nodes, we might need to decide on more than two boundary conditions. In the case of models with no pendant nodes, we do not necessarily need any boundary conditions.

A non-pendant node may have multiple boundary conditions. Take the example of a rectangular plate which is clamped at the left and right edges and free to move at the top and bottom edges. In this case the corners of the plate will have a ”free” boundary condition in the top-bottom direction and a ”clamped” boundary condition in the left-right direction. For our graph models we will similarly

have to decide on a boundary condition for each direction, however, our models may have more than two directions. If a node in our model has boundary conditions we call it an *edge node*.

The number of boundary conditions to be decided for an edge node will be the same as its connectivity. For example, if we consider the middle node in Figure 1. b) to be an edge node, we need to implement boundary conditions for the directions of each of the three connected strings.

Two commonly used boundary conditions are [1]

$$u = 0 \qquad \text{(Dirichlet)}, \tag{24}$$

$$u_x = 0 \qquad \text{(Neumann)}. \tag{25}$$

For simplicity we will assume for a given node, all directions will have the same boundary condition.

To implement the Dirichlet condition we introduce a virtual node with a constant value 0 for each direction. This gives us the update rule

$$\delta_{\Delta\mathbb{U}} u_l^n = \frac{1}{h^2}\left(\sum_{\overset{*}{u}\in\mathbb{U}_l} \overset{*}{u} - 2\left(|\mathbb{U}_l|\right) u_l\right). \tag{26}$$

Implementation of the Neumann condition involves setting $\delta_{t\cdot} = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right) = 0$ for each direction, which amounts to creating a virtual node with the same value as the real node of that direction, see [1, chapter 5.2.8] for further details. Doing this we get an update rule which looks like

$$\delta_{\Delta\mathbb{U}} u_l^n = \frac{1}{h^2}\left(2\sum_{\overset{*}{u}\in\mathbb{U}_l} \overset{*}{u} - 2|\mathbb{U}_l| u_l\right). \tag{27}$$

There are lots of other options for boundary conditions apart from Neumann and Dirichlet, see e.g. [9, chapter 19] for some choices.

## 7. COMPUTATIONAL COMPLEXITY

Knowing the computational complexity of a given model is critical if one wishes to run large models or run models in real time. Since graph theory is a well studied area of computer science a lot of existing material will cover complexity analysis similar to the one of this section (see e.g. [16]). Despite this we will still go through a basic complexity analysis specific to the topic at hand.

The complexity of a node will depend on the amount of connections to it. Evaluating Equation (16) for a pendant node will take just one operation when disregarding the division by $h^2$. Connected notes will take $|\mathbb{U}|+1$ operations: $|\mathbb{U}-1|$ operations for the summation and two for the last addition and multiplication.

If we disallow cycles, the worst case model is then the string topology model because it has the lowest amount of pendant nodes. The complexity of the string model is $3(n-2)+2 = \mathcal{O}(n)$. The acyclic model with the highest amount of pendant nodes has one node connected to every other node in the model, this has complexity $n+(n-1) = \mathcal{O}(n)$. Thus any acyclic will have complexity $\mathcal{O}(n)$.

If we allow cycles we may increase the complexity of our model. The most complex model is the one where every node connects to every other node, giving us a complexity of $n(n+1) = \mathcal{O}(n^2)$, which is thus the upper bound of any model we can build.

## 8. HARMONIC CONTENT

Finding the harmonic content of a given model can be done using common methods for computing the natural modes of a mass-spring system [1, chapter 3]. This method is completely standard and should be found in any good textbook on the subject, but is presented here specific to the topic at hand.

A given mass spring system can be described using the equation

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}, \tag{28}$$

where $\mathbf{M}$ encodes the masses, $\mathbf{K}$ the spring constants and $\mathbf{u}$ the displacements of the masses.

We then assume that Equation (28) has a solution of the form

$$\mathbf{u} = \mathbf{U}e^{i\omega t}, \tag{29}$$

which we plug back into Equation (28) to get

$$\left(\mathbf{K} - \omega^2\mathbf{M}\right)\mathbf{U}e^{i\omega t} = \mathbf{0} \tag{30}$$

and since $e^{i\omega t} \neq 0$ we have that

$$\left(\mathbf{K} - \omega^2\mathbf{M}\right)\mathbf{U} = \mathbf{0}. \tag{31}$$

Multiplying through by $\mathbf{M}^{-1}$ we get

$$\left(\mathbf{M}^{-1}\mathbf{K} - \omega^2\mathbf{I}\right)\mathbf{U} = \mathbf{0}, \tag{32}$$

which is analogous to the canonical form of the eigenvalue problem

$$\left(\mathbf{A} - \lambda\mathbf{I}\right)\mathbf{v} = \mathbf{0} \tag{33}$$

when setting $\mathbf{A} = \mathbf{M^{-1}K}$ and $\lambda = \omega^2$.

Therefore finding the eigenvalues of $\mathbf{M}^{-1}\mathbf{K}$ will give us the frequencies of the natural modes of the system, the corresponding eigenvectors $\mathbf{v}$ will be the shape of the given mode.

Since our models are characterized by the wave speed $c^2$ and the topology of the graph, we need a way deriving $\mathbf{M}$ and $\mathbf{K}$ from these.

Using the definitions from Section 3. and selecting $A = 1$ and $\rho = 1$ we get

$$\kappa = \frac{c^2}{h} \quad \text{and} \quad m = h. \tag{34}$$

Since all nodes in our system have the same mass we have

$$\mathbf{M} = h\mathbf{I}. \tag{35}$$

The shape of $\mathbf{K}$ will depend on the topology of the graph. For example, considering only the center node $u_L$ of Figure 3. we get matrices which look like

$$\mathbf{K}\mathbf{u} = \begin{bmatrix} & & \cdots & \\ \kappa & -3\kappa & \kappa & \kappa \\ & & \cdots & \\ & & \cdots & \end{bmatrix} \begin{bmatrix} u_{L-1} \\ u_L \\ \hat{u_1} \\ \check{u_1} \end{bmatrix}. \tag{36}$$

This process of building the $\mathbf{K}$ matrix can and should be done using software for models of large sizes.

## 9. EXTENSIONS

So far we have looked only at the case of building models from the 1D wave equation. It is, however, possible to use the same principles for other acoustic objects.

For example, we can take a linear bar model [1]

$$\frac{d^2}{dt^2}u = -\kappa^2 \frac{d^4}{dx^4}u, \tag{37}$$

which can be discretized as

$$\delta_{tt}u_l^n = -\kappa^2 \delta_{xx}\delta_{xx}u_l^n, \tag{38}$$

after which we apply Equation (10) instead of $\delta_{xx}$ to get

$$\delta_{tt}u_l^n = -\kappa^2 \delta_{\Delta\mathbb{U}}\delta_{\Delta\mathbb{U}}u_l^n, \tag{39}$$

from which we can isolate $u_l^{n+1}$ to get our update rule.

For clarity, the operator $\delta_{\Delta\mathbb{U}}\delta_{\Delta\mathbb{U}}$ is evaluated as

$$\delta_{\Delta\mathbb{U}}\delta_{\Delta\mathbb{U}}u_l^n = \frac{1}{h^2}\left(\sum_{\overset{*}{u}\in\mathbb{U}_l} \delta_{\Delta\mathbb{U}}\overset{*}{u} - |\mathbb{U}_l|\delta_{\Delta\mathbb{U}}u_l^n\right). \tag{40}$$

One can also extend models by adding dampening. For example we can build 1D wave equation based model with dampening by starting with the equation

$$u_{tt} = c^2 u_{xx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}, \tag{41}$$

where $\sigma_0$ is a constant controlling frequency independent loss and $\sigma_1$ controlling frequency dependent loss [2], and discretizing it like

$$\delta_{tt}u_l^n = c^2\delta_{\Delta\mathbb{U}} - \sigma_0\delta_t u_l^n + 2\sigma_1\delta_t\delta_{\Delta\mathbb{U}}u_l^n. \tag{42}$$

## 10. IMPLEMENTATION

A reference implementation in MATLAB is available online [2], providing a class for building models by creating strings, connecting them, and adding boundary conditions, after which one can compute the $\delta_{\Delta\mathbb{U}}$ and $\delta_{\Delta\mathbb{U}}\delta_{\Delta\mathbb{U}}$ operators.

Using the main class, an implementation of a 1D wave equation based model is provided, with and without dampening. An implementation of the linear bar model is also provided, again with and without dampening, showcasing the use of the $\delta_{\Delta\mathbb{U}}\delta_{\Delta\mathbb{U}}$ operator.

The repository also contains a work in progress — though usable — interactive GUI application written in C++ / JUCE, which will serve as a test bed for various model topologies, extensions and exiters.

## 11. FUTURE WORK

The most pressing issue for the practical utility of the current research is to show stability conditions for a given graph. This could likely be done using the energy method. Tree-like graphs seem to have excellent stability conditions comparable to the 1D wave equation, which is stable whenever $ck/h \leq 1$ [1, Chapter 6]. However, since we can build meshes equal to the 2D wave equation, there must also be a case where the stability condition is $ck/h \leq 1/\sqrt{2}$ [1, Chapter 11].

Some feature of the topology of a model must be the determining factor for the stability condition. Finding a condition such that the stability of a given graph can be known before evaluating it is of vital importance if algorithms such as these should ever be used by non-experts.

Like other finite difference models, we need a way of exciting the system. Many choices are available ranging from simple initial conditions, to advanced bow, hammer or reed excitation (see e.g. [17]). Any exciter applicable to the 1D wave equation should be applicable to graph based models.

More work can be done investigating the frequency content of graphs. For example, how does the relationship between the lengths of the string in the branching topology affect the modal frequencies? and what happens when we introduce cycles into our model? How do models behave when built using stiff strings or bar models?

Throughout this paper we have considered junctions between strings of equal stiffness. One could derive rules similar to the ones in this paper, but for strings with differing stiffness, which would lead to even more ways of building graphs.

Lastly, there is of course a lot of time to be spent exploring the various timbres and artistic uses of graph based physical models, and related to that, new interfaces for controlling and performing with such models.

## 12. CONCLUSION

In this paper we have explored some of the fundamental concepts of constructing and analysing graph based physical models for sound synthesis.

Starting with a review of the 1D wave equation and one of its derivations using mass-spring networks, we showed how to build a second order difference operator applicable to the end of a string which branches out into $N$ other strings. Using this we are able to construct any kind of cyclic and acyclic graph.

When a model is built without string segments longer than one, we call it dense. We showed that using our new difference operator, we can build a model which is equivalent to a discretization of the 2D wave equation. We then created a grid using hexagons and found that it was superior to the rectangular grid with regards to stability and computational complexity.

Like other finite-difference schemes we needed to decide on some boundary conditions for the edge nodes in our models. We reviewed the Neumann and Dirichlet boundary conditions and showed how to implement them for graph models.

By reasoning about the number of pendant nodes in a graph, we demonstrated that acyclic models have a computational complexity of $\mathcal{O}(N)$, and that cyclic graphs have a worst case complexity of $\mathcal{O}(N^2)$.

Using common methods for analyzing vibrating systems, we showed how to use the parameters of our model to set

[2] https://github.com/PelleJuul/graph-physical-models

up a linear system in canonical eigenvalue problem form, from which we can compute the modal frequencies and shapes.

Some extensions to our models were examined, including how to apply our results to a linear bar models and how to add dampening to a system.

Lastly we discussed topics for future research including a call for a more rigid mathematical analysis of the models, experiments with various excitation mechanisms, investigations of non-homogeneous models, and a wish for future artistic and interaction related endeavors related to graph based physical modeling.

**Acknowledgments**

The authors would like to express sincere gratitude to Silvin Willemsen and Nikolaj Andersson for their feedback, ideas and enthusiasm for this project, as well as a general thanks to all of the staff and students at the SMC master's programme at Aalborg University Copenhagen, for building a great environment for learning and experimentation. This work is partially supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

## 13. REFERENCES

[1] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley and Sons, 2009.

[2] ——, "A modular percussion synthesis environment," in *Proc. Int. Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, 2009.

[3] A. L. Claude Cadoz and J. L. Florens, "Cordis-anima: A modeling and simulation system for sound and image synthesis: The general formalism," in *Computer Music Journal Vol. 17, No. 1*, 1993, pp. 19–29.

[4] G. Eckel, F. Iovino, and R. Causs, "Sound synthesis by physical modelling with modalys," in *Proceedings of the International Symposium of Music Acoustics*, 1995.

[5] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic modeling using the digital waveguide mesh," 2007.

[6] D. Rocchesso and F. Fontana, "The sounding object," 2003.

[7] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations, Second Editon*. Society for Industrial and Applied Mathematics, 2004.

[8] R. R. Rosales, "Force-directed drawing algorithms," 2001.

[9] S. Bilbao, B. Hamilton, R. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," in *Springer Handbook of Musical Acoustics*, 2018, pp. 249–384.

[10] B. Hamilton and S. Bilbao, "Hexagonal vs. rectilinear grids for explicit finite difference schemes for the two-dimensional wave equation," in *21st International Congress on Acoustics*, 2013.

[11] J. O. Smith III, "On the equivalence of the digital waveguide and finite difference time domain schemes," 2004.

[12] M. Karjalainen, P. Huang, and J. O. S. III, "Digital waveguide networks for room response modeling and synthesis," in *Proc. of the 118th AES conference*, 2005.

[13] J. O. Smith III, "Aspects of digital waveguide networks for acoustic modeling applications," 1997.

[14] G. Essel and P. R. Cook, "Banded waveguides on circular topologies and of beating modes: Tibetan singing bowls and glass harmonicas," 2002.

[15] D. T. Murphy, "Digital waveguide mesh topologies in room acoustics modelling," 2001.

[16] S. G. Kobourov, "Force-directed drawing algorithms," 2004.

[17] F. Avanzini, M. Rath, D. Rocchesso, and L. Ottaviani, "The sounding object," D. Rocchesso and e. Federico Fontana, Eds., 2003, ch. 8.